## REMARKS

With regard to the claim objection in section one of the Office Action, the claims amended above have been changed to double line spacing.

Claims 1-27 have been rejected as anticipated under 35 USC 102(e) by Adelman, U.S. patent 6,078,957. In response to that rejection, the claims have been amended as described below.

The object of the invention is to present a flexible method and arrangement for handling dynamic state information in nodes of a network element cluster. A further object is to present such a method and arrangement for handling dynamic state information which is effective and allows flexible transferring of packet data connections between nodes of a network element cluster at the same time. An important feature of the invention is the segregation of state data in each node in a cluster into two segments: a first segment that only pertains to sets of data packets being handled by that node; and a second segment that pertains to sets of data packets being handled by other nodes in the cluster. This improves performance because when a packet arrives at any particular node, only the state data for packets being handled by that node needs to be searched which greatly speeds up the search. The existence in each node of state data which pertains to sets of packets being handled by other nodes allows load balancing to be accomplished easily. Load balancing means sets of data packets being handled by a node which is overloaded to be transferred to another node for processing. This can be easily and rapidly done in the invention by transferring state data which pertains to new sets of data packets to be handled by a first node which were previously being handled by a second node from the second store of state data in the first node to the first store of state data in the first node.

The invention finds application in network element clusters which together handle data traffic, typically to terminate virtual private networks. Clusters enable reliability by allowing shifting of traffic away from failed nodes. Clusters also improve performance by dividing the traffic load among multiple machines.

Data packets arriving at a cluster are distributed to different nodes of the cluster. A set of packets may refer, for example, to one connection or to a communication session comprising multiple connections. Likewise, the sets of data packets handled by a node may refer to a secure tunnel or to some other type of communication.

"Stateful" handling of data packets refers to a situation where a first data packet of a set of data packets is handled without any information about the history of data packets as in stateless handling, and consequently, information about the respective set of data packets is stored in the network element for handling the rest of the data packets belonging to the same set of data packets. This information represents the state of the set of data packets and is referred to as state information and is stored in a data structure herein referrred to as a state data structure. The part of the state data structure that is related to one set of data packets is called an entry.

Since the state information is required for handling the data packets, transferring a connection from one node to another node would mean transferring also the state information. Therefore, in the prior art solutions, each node maintains effectively all state data entries related to all sets of data packets in the cluster at a given moment of time. Synchronization of the state information between the nodes can be carried out in order to enable the transfer of the connection in a fail-over situation, but the scalability of the cluster is lost. On the other hand, if the state information is not synchronized between nodes, the scalability is improved, but the transfer of the connection in a fail-over situation is not possible.

The further disadvantage of the prior art solutions is that since each node needs to maintain information about all sets of data packets handled in said cluster, the size of the required state data structure may be large. This is the case especially if there are many nodes in the cluster and consequently a large number of connections handled in the cluster. The time needed for finding a match for a data packet in a state data structure clearly increases as the size of the state data structure increase, which deteriorates the performance of the nodes. At the same time, the resources for maintaining the state data structure may be limited.

In accordance with the teachings of the invention, the second common data structure contains full copies of all state information needed for handling sets of data packets handled in other nodes of the network element cluster in order to enable flexible transferring of connections between nodes. A first node-specific data structure in each node stores only the state data structure entries needed to handle the data packets for the connections assigned to the node. This much smaller set of data structure entries speeds up the lookup of the state information when a packet from a connection assigned to the node is encountered because there are fewer state information entries to search for the one or more entries pertinent to said connection of which said packet is a part. This maintains high performance.

The possibility to reliably maintain these state data structures is provided especially by knowledge of how different sets of data packets are distributed within the nodes of the network element cluster. Each node contains in second common data structure all the state information pertinent to the connections being handled by the other nodes in the cluster plus a copy of the state information that is in the first node specific data structure of the node. This allows rapid redistribution of connections from one node to another for load balancing and reliability purposes. This is because the state data structures needed in the first node specific data structure of the node to which the connection has been reassigned can simply be transferred from the node's second common data structure to its first node-specific data structure, and there is no need for transfers of state data information between nodes of the cluster.

When a distribution of at least one set of data packets is dynamically changed by assigning it from at least one other node to the node in question, the node is provided with a respective changed distribution information. This new distribution information updates a distribution function which defines the data packets which said node is assigned to process. At the same time, in response to the changed distribution information, the node also selects the state information entries of said at least one re-distributed set of data packets from the second common data structure and transfers them to the first node-specific data structure of the node to prepare the node to handle the data packets of the connection or connections newly assigned to it. Thus, the state information is available in the node-specific data structure immediately after the distribution is changed so the node is immediately ready to handle the newly assigned set of data packets before any data packet of the new set is received in the node.

When the load distribution decision is taken into account in the synchronization of the state information in accordance with the present invention, the synchronization can be carried out so that the size of the active data structure (the first node-specific data structure) corresponds to that of the non-synchronized situation while achieving the benefits of the synchronization.

Adelman discloses n IP network cluster, such as VPN tunnel-server cluster, or a firewall cluster, etc. The traffic load is distributed among the member units of the cluster. As shown in Figure 5, and described in Col. 6, lines 21-29, each cluster member contains a work assignment table 515 which contains the session work-unit hash numbers and the cluster member ID assigned to that work unit; an application state table 517 for this cluster member; and a similar application state table 519 for the other members of the cluster.

In Adelman, the application state table for the other members of the cluster contains only subsets of the state information needed for handling the packets in the other nodes (Col. 12, lines 35 to 43). Special calculation operations are needed when these subsets are taken in the use in a new node.

Adelman fails to teach or suggest how the application state tables are actually used. Adelman does teach however at Col. 11, line 63 to Col. 12, lines 61 and Figure 7 that the state of a TCP message constantly changes and it is not practical for a cluster member to transfer all of this TCP state to all of the other members of the cluster each time the state is changed. Figure 7 shows that the state of a TCP message is quite an extensive collection of data. Transferring all this state data would require an extensive amount of storage and processing time and would essentially double the traffic to the members of the cluster. The ability of themember units to maintain the state of these incoming messages is critical to their ability to handle the failure of a member unit without requiring a reset of the message session. Figure 7 depicts the way the Adelman patent divides the state up into three portions: an initial state which only needs to be sent once; an essential state which is sent upon each state change; and a calculable state which is calculable from the essential state and is not sent when the state changes.

A person skilled in the art would not find every element of the claims as amended in the Adelman patent as is needed to support an anticipation rejection. For example, there is no teaching in Adelman of transferring state information from the common data structure of a first node to the node-specific data structure of the first node when the data packets of a connection of a second node are assigned to the first node, and there is no teaching of the specific types of state information which are stored in the node specific data structure and the common data structure.

A person skilled in the art would not have modified the system of Adelman so as to achieve the present invention as claimed. The skilled person would have used the storage capacity benefit offered by the approach of Adelman, so that upon receiving a packet belonging to a work set owned by the node, the node first searches the classification state table of that node, and if the entry is not found, the node perform a second search in the application state table for the other members of the cluster. The node then calculates the full state information from the subset of the state information, and stores the calculated information in the node-specific application state table.

On the basis of the above arguments and the amendments to the claims, applicants submit that the amended claims are not anticipated by the Adelman patent.

Respectfully submitted,

Dated: June 23, 2005

[illegible]

Ronald Craig Fish
Reg. No. 28,843
Tel 408 866 4777

I hereby certify that this correspondence is being deposited with the United States Postal Service as First Class Mail, postage prepaid, in an envelope addressed to: Commissioner for Patents , P.O. Box 1450, Alexandria, Va. 22313-1450.
on 6/23/05
(Date of Deposit)

[illegible]
Ronald Craig Fish, President
Ronald Craig Fish, a Law Corporation

Reg. No. 28,843